Bharat Intern

Name - Shiva D. Mehenge

In [1]:
```
# Import Packages and Librarys
```

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
# Pandas use for the Data Wrangling and Manipulation

df = pd.read_csv("E:\\CSV Data\\WA_Fn-UseC_-HR-Employee-Attrition.csv")
df
```

Out[2]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	E
0	41	Yes	Travel_Rarely	1102	Sales	1	2	
1	49	No	Travel_Frequently	279	Research & Development	8	1	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	
4	27	No	Travel_Rarely	591	Research & Development	2	1	
...	...	...	...	...	...	...	...	...
1465	36	No	Travel_Frequently	884	Research & Development	23	2	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	
1468	49	No	Travel_Frequently	1023	Sales	2	3	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	

1470 rows × 35 columns

In [29]:
```
df.head(10)
```

Out[29]:

	YearsAtCompany	YearsInCurrentRole	YearsSinceLastPromotion	YearsWithCurrManager	Attriti
0	5	2	1	3	
1	3	1	2	1	
2	7	3	3	5	
3	2	1	1	1	
4	4	2	2	2	
5	8	3	3	4	
6	6	2	2	3	
7	1	1	1	1	
8	3	1	1	2	
9	2	2	2	3	

In [4]:
```
df.tail(3)
```

Out[4]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	E
1467	27	No	Travel_Rarely	155	Research & Development	4	3	
1468	49	No	Travel_Frequently	1023	Sales	2	3	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	

3 rows × 35 columns

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [6]:
```
df["Age"].value_counts()
```

Out[6]:
```
35    78
34    77
36    69
31    69
29    68
32    61
30    60
33    58
38    58
40    57
37    50
27    48
28    48
42    46
39    42
45    41
41    40
26    39
44    33
46    33
43    32
50    30
25    26
24    26
49    24
47    24
55    22
51    19
53    19
48    19
54    18
52    18
22    16
56    14
23    14
58    14
21    13
20    11
59    10
19     9
18     8
60     5
57     4
Name: Age, dtype: int64
```

In [22]:
```
df["Education"].value_counts()
```

Out[22]:
```
3    572
4    398
2    282
1    170
5     48
Name: Education, dtype: int64
```

In [ ]:

Descriptive Statistics

In [8]:
```
df.nunique()
```

Out[8]:
```
Age                         43
Attrition                    2
BusinessTravel               3
DailyRate                  886
Department                   3
DistanceFromHome            29
Education                    5
EducationField               6
EmployeeCount                1
EmployeeNumber            1470
EnvironmentSatisfaction      4
Gender                       2
HourlyRate                  71
JobInvolvement               4
JobLevel                     5
JobRole                      9
JobSatisfaction              4
MaritalStatus                3
MonthlyIncome             1349
MonthlyRate               1427
NumCompaniesWorked          10
Over18                       1
OverTime                     2
PercentSalaryHike           15
PerformanceRating            2
RelationshipSatisfaction     4
StandardHours                1
StockOptionLevel             4
TotalWorkingYears           40
TrainingTimesLastYear        7
WorkLifeBalance              4
YearsAtCompany              37
YearsInCurrentRole          19
YearsSinceLastPromotion     16
YearsWithCurrManager        18
dtype: int64
```

In [9]:
```
df.values
```

Out[9]:
```
array([[41, 'Yes', 'Travel_Rarely', ..., 4, 0, 5],
       [49, 'No', 'Travel_Frequently', ..., 7, 1, 7],
       [37, 'Yes', 'Travel_Rarely', ..., 0, 0, 0],
       ...,
       [27, 'No', 'Travel_Rarely', ..., 2, 0, 3],
       [49, 'No', 'Travel_Frequently', ..., 6, 0, 8],
       [34, 'No', 'Travel_Rarely', ..., 3, 1, 2]], dtype=object)
```

In [10]:
```
df.describe()
```

Out[10]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNu
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.00
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.86
std	9.135373	403.509100	8.106864	1.024165	0.0	602.02
min	18.000000	102.000000	1.000000	1.000000	1.0	1.00
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.25
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.50
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.75
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.00

8 rows × 26 columns

Data Cleaning

In [11]:
```
df.isna()
```

Out[11]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Ed
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
1465	False	False	False	False	False	False	False	
1466	False	False	False	False	False	False	False	
1467	False	False	False	False	False	False	False	
1468	False	False	False	False	False	False	False	
1469	False	False	False	False	False	False	False	

1470 rows × 35 columns

In [12]:
```
df.isna().sum()
```

Out[12]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

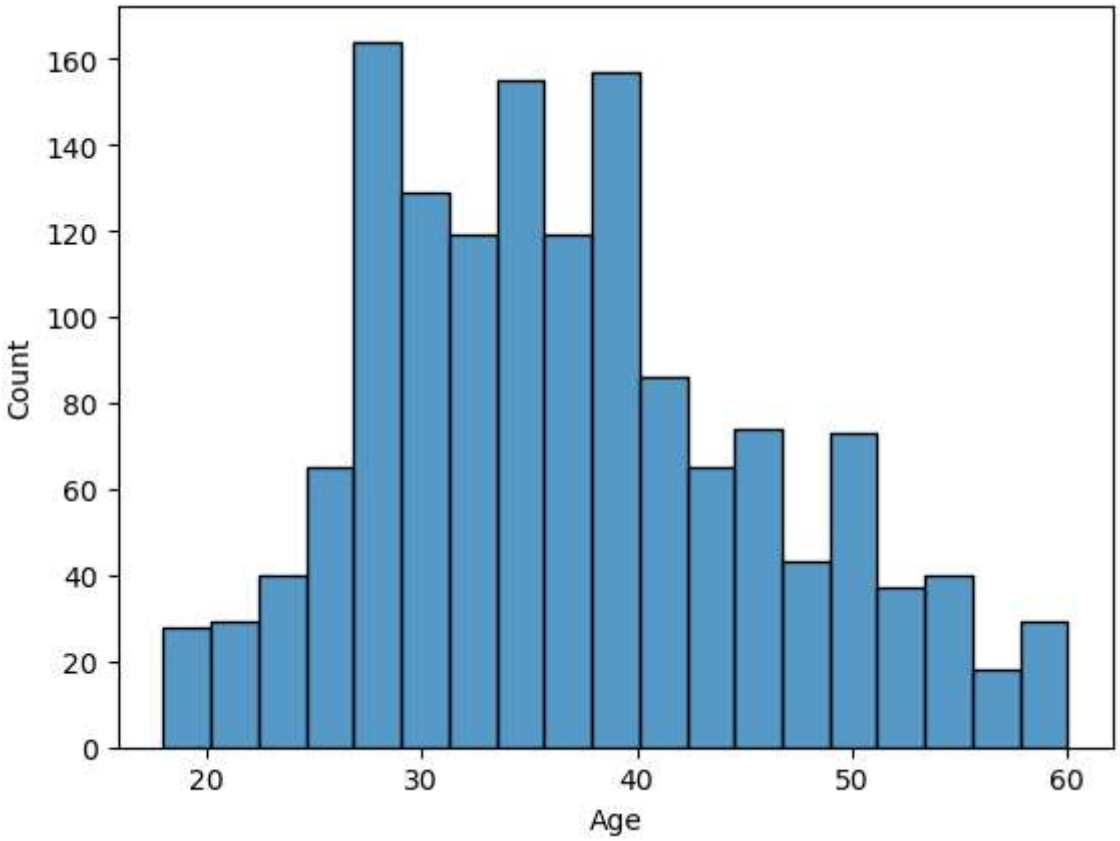
In [13]:
```

```

In [ ]:
```

```

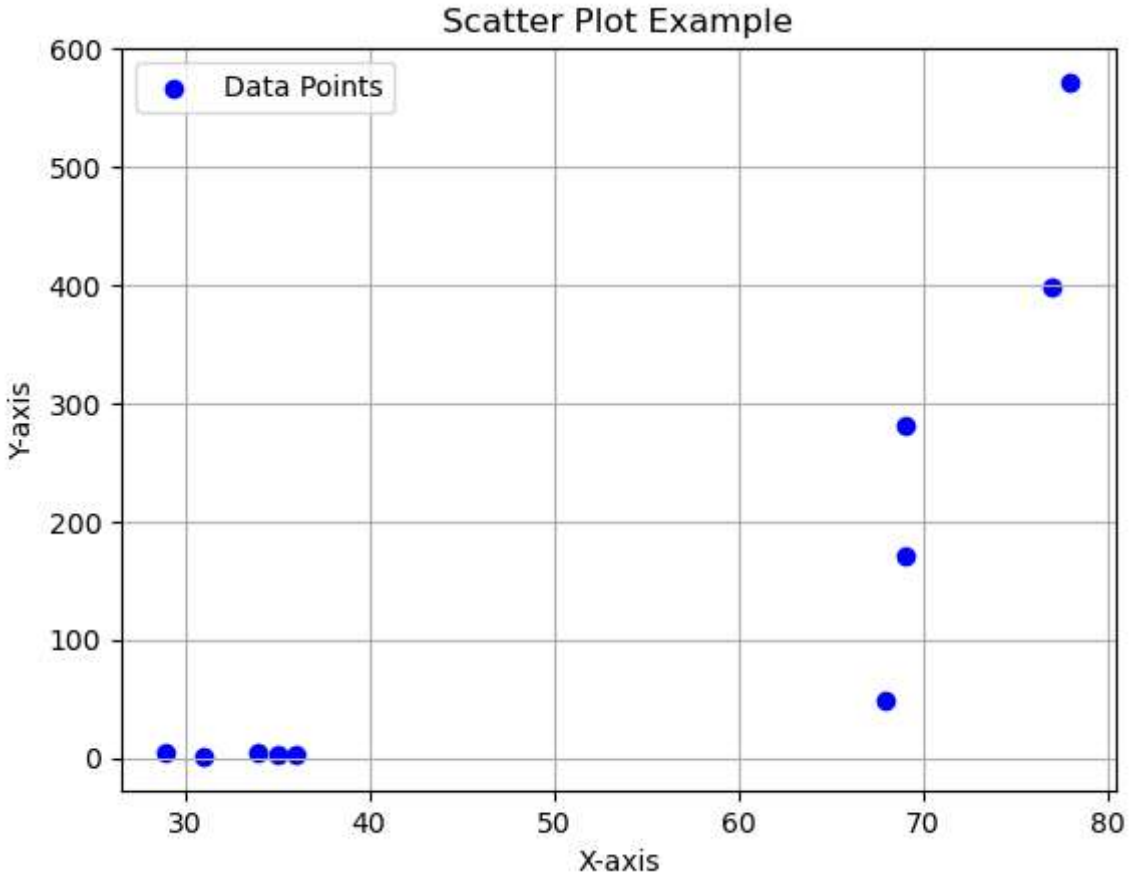
Data Visualization

In [15]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [16]:
```
sns.histplot(x='Age' ,  data= df,)
plt.show()
```



In [23]:
```
import matplotlib.pyplot as plt

# Sample data for the scatter plot
x = [35,78,34,77,36,69,31,69,29,68]
y = [3,572,4,398,2 ,282,1 ,170,5 ,48]

# Create the scatter plot
plt.scatter(x, y, color='blue', marker='o', label='Data Points')
plt.title('Scatter Plot Example')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.grid(True)
plt.legend()
plt.show()
```

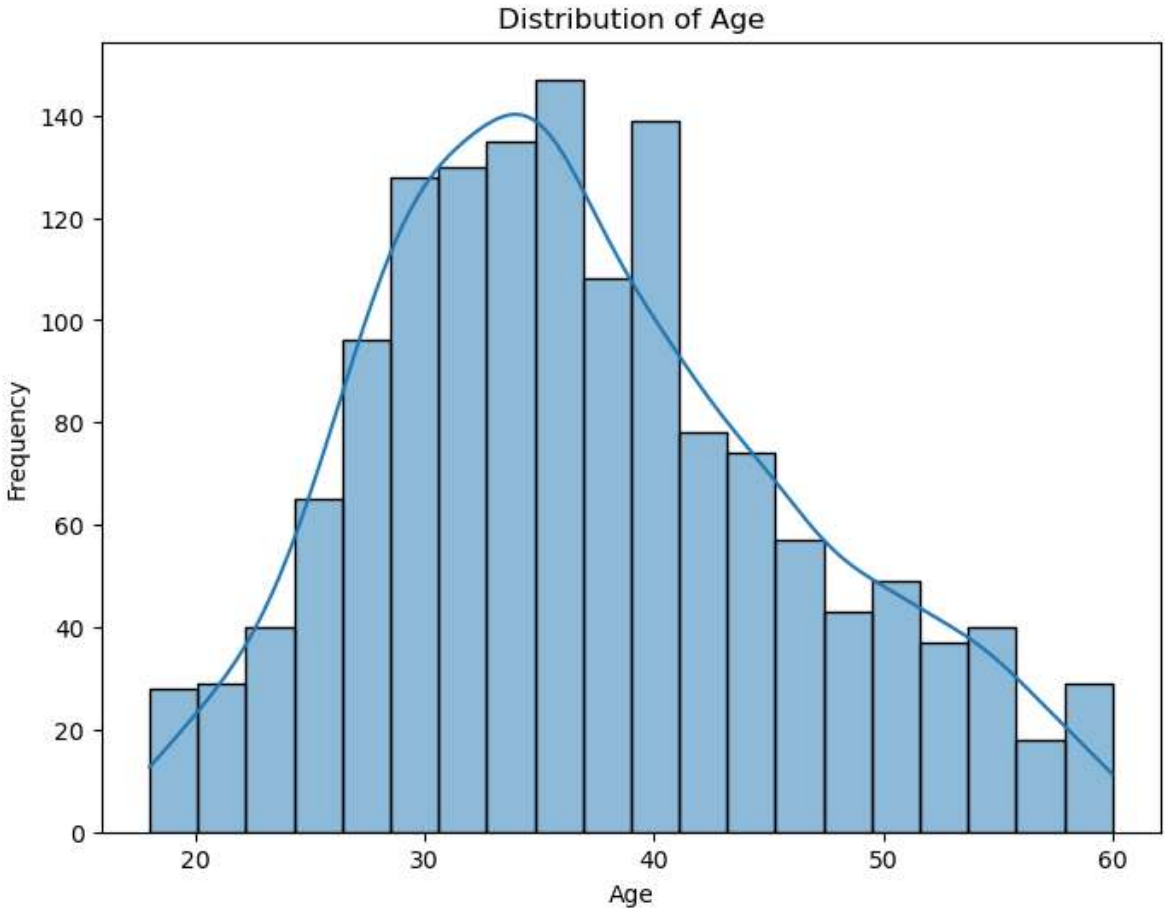


In [42]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

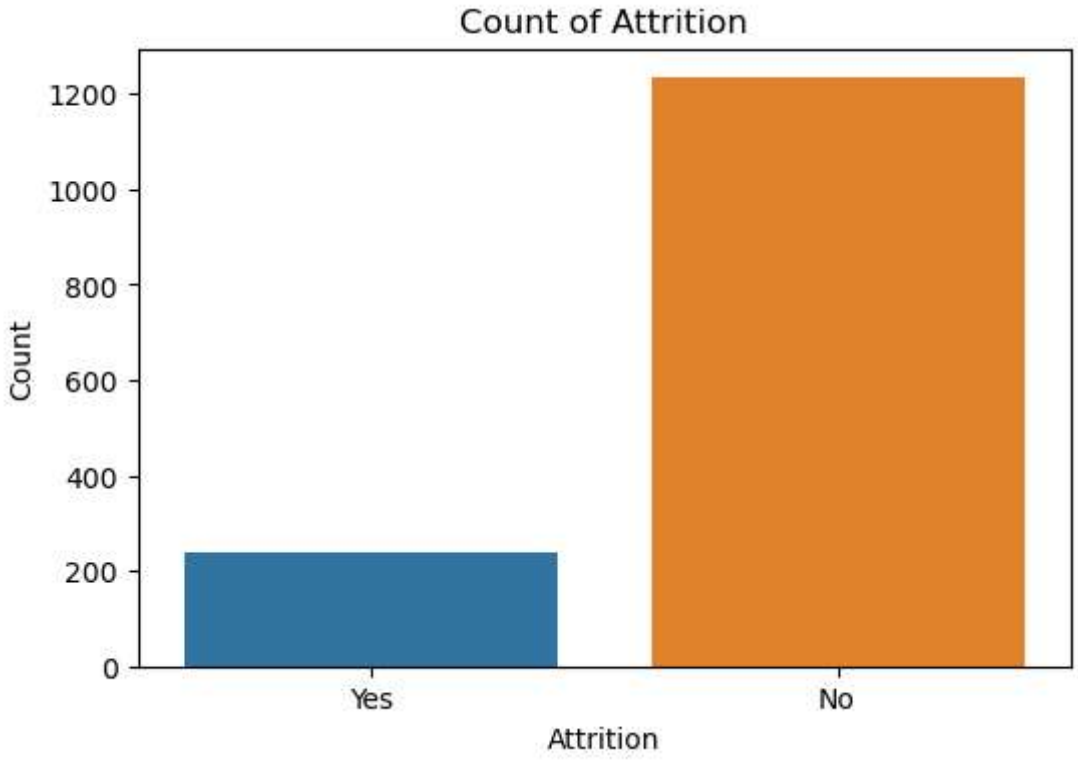
In [43]:
```
import seaborn as sns
```

In [38]:
```
# Load the data (replace 'data.csv' with the actual filename)
data = pd.read_csv("E:\\CSV Data\\WA_Fn-UseC_-HR-Employee-Attrition.csv")
```

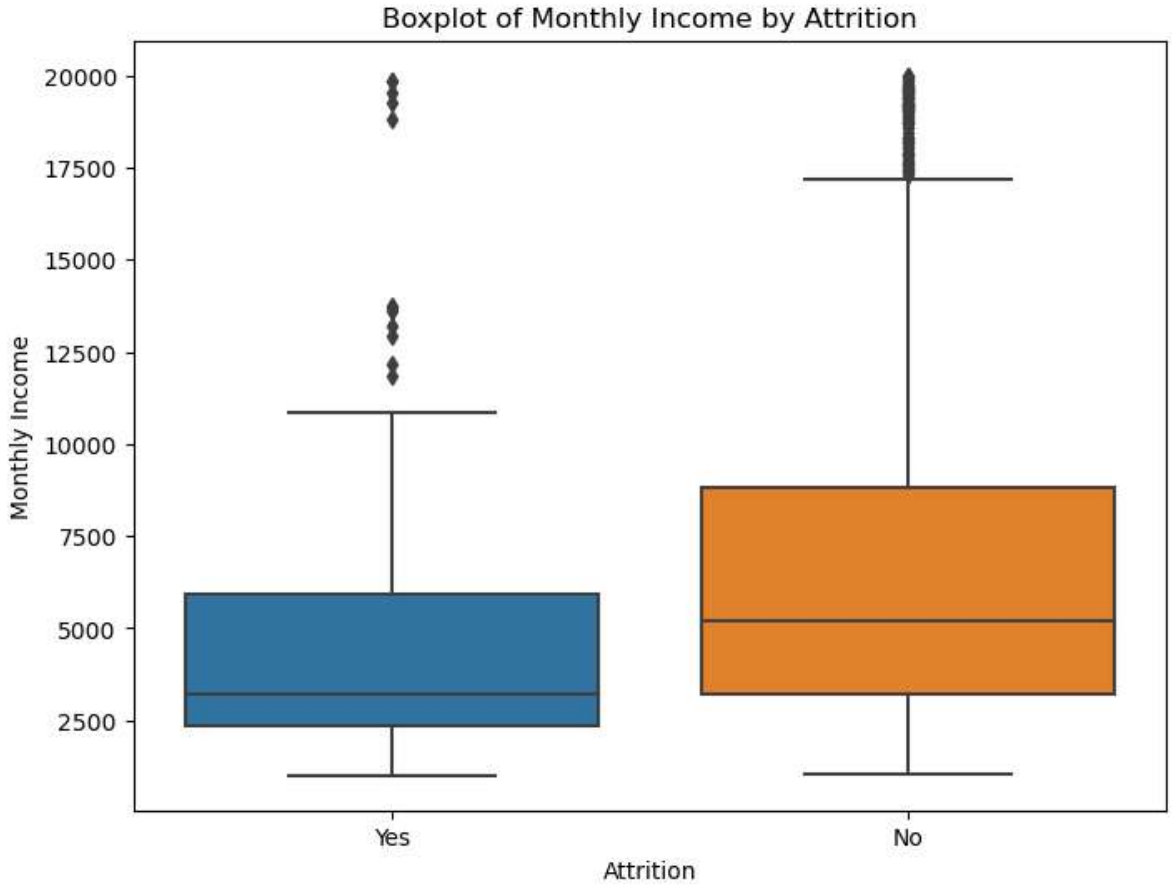
In [34]:
```
# Plot the distribution of 'Age'
plt.figure(figsize=(8, 6))
sns.histplot(data['Age'], bins=20, kde=True)
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Distribution of Age')
plt.show()
```



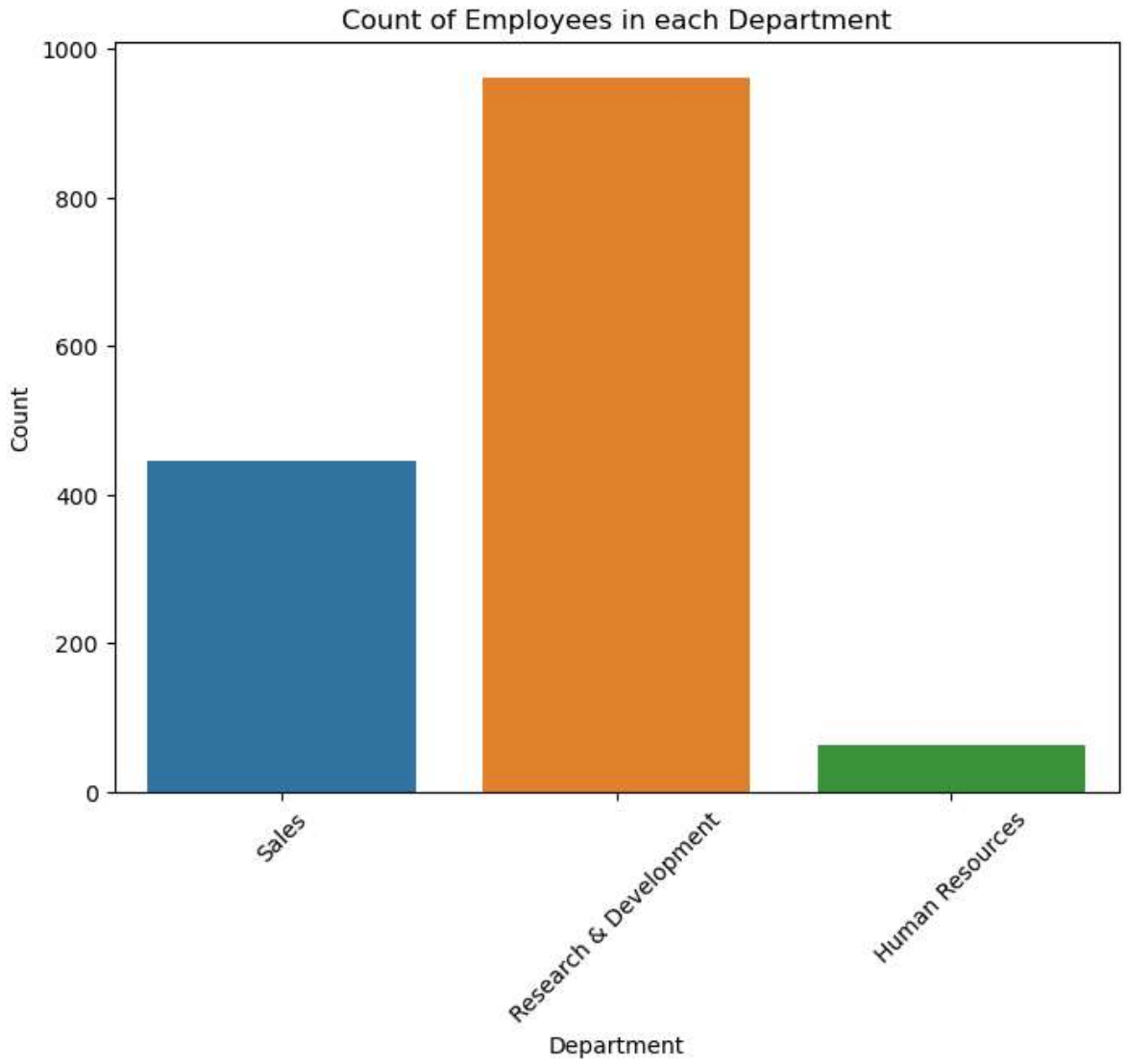
In [35]:
```
# Plot the count of 'Attrition'
plt.figure(figsize=(6, 4))
sns.countplot(x='Attrition', data=data)
plt.xlabel('Attrition')
plt.ylabel('Count')
plt.title('Count of Attrition')
plt.show()
```



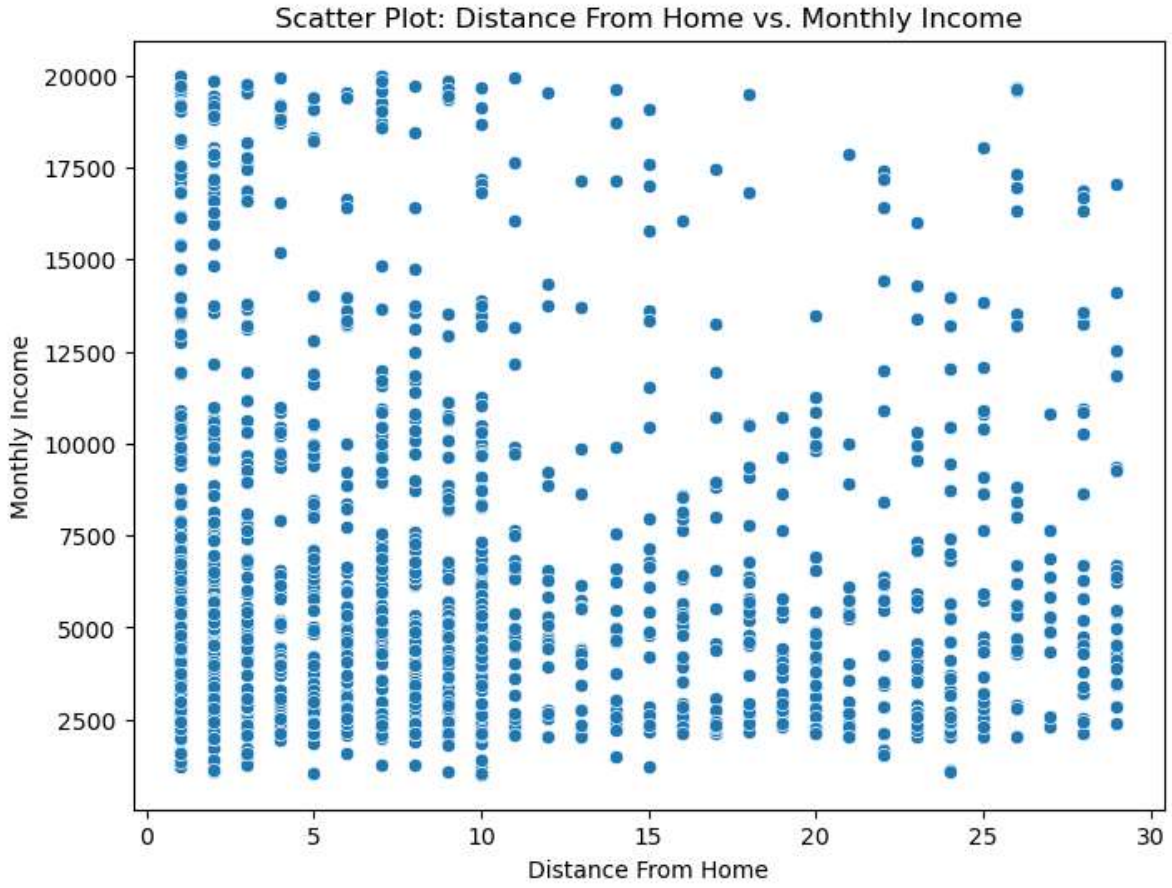
In [36]:
```
# Plot the boxplot for 'MonthlyIncome' by 'Attrition'
plt.figure(figsize=(8, 6))
sns.boxplot(x='Attrition', y='MonthlyIncome', data=data)
plt.xlabel('Attrition')
plt.ylabel('Monthly Income')
plt.title('Boxplot of Monthly Income by Attrition')
plt.show()
```



In [37]:
```
# Plot the bar chart for 'Department'
plt.figure(figsize=(8, 6))
sns.countplot(x='Department', data=data)
plt.xlabel('Department')
plt.ylabel('Count')
plt.title('Count of Employees in each Department')
plt.xticks(rotation=45)
plt.show()
```



In [39]:
```
# Plot the scatter plot for 'DistanceFromHome' vs. 'MonthlyIncome'
plt.figure(figsize=(8, 6))
sns.scatterplot(x='DistanceFromHome', y='MonthlyIncome', data=data)
plt.xlabel('Distance From Home')
plt.ylabel('Monthly Income')
plt.title('Scatter Plot: Distance From Home vs. Monthly Income')
plt.show()
```

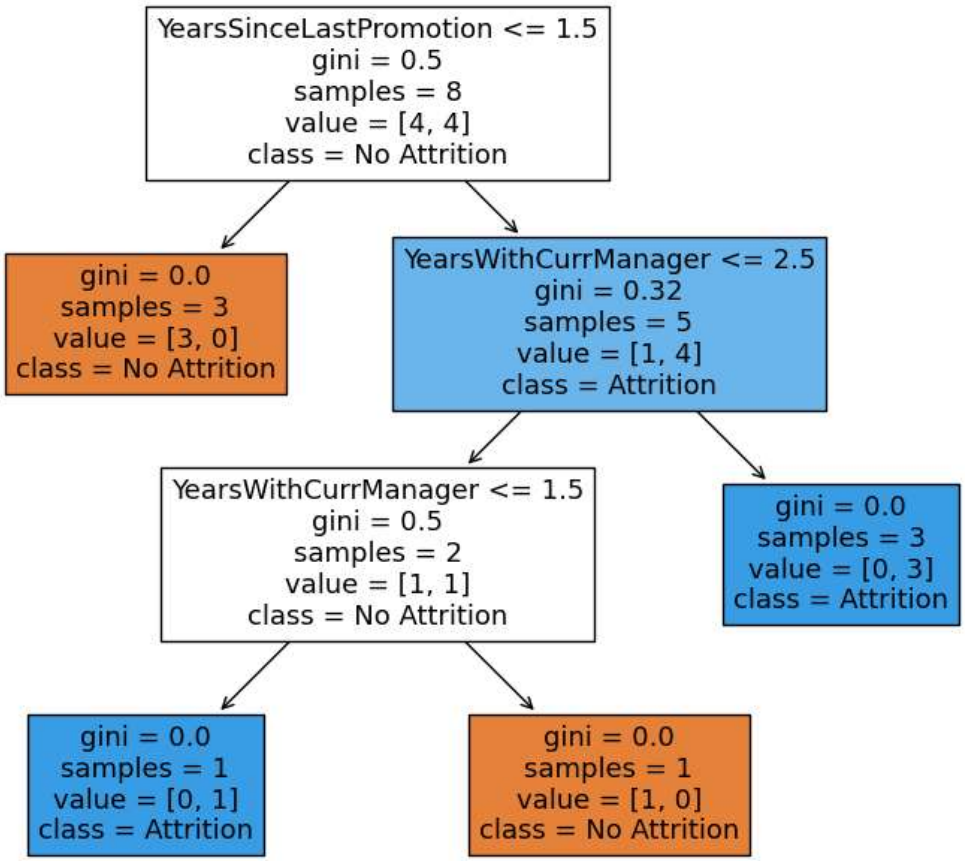


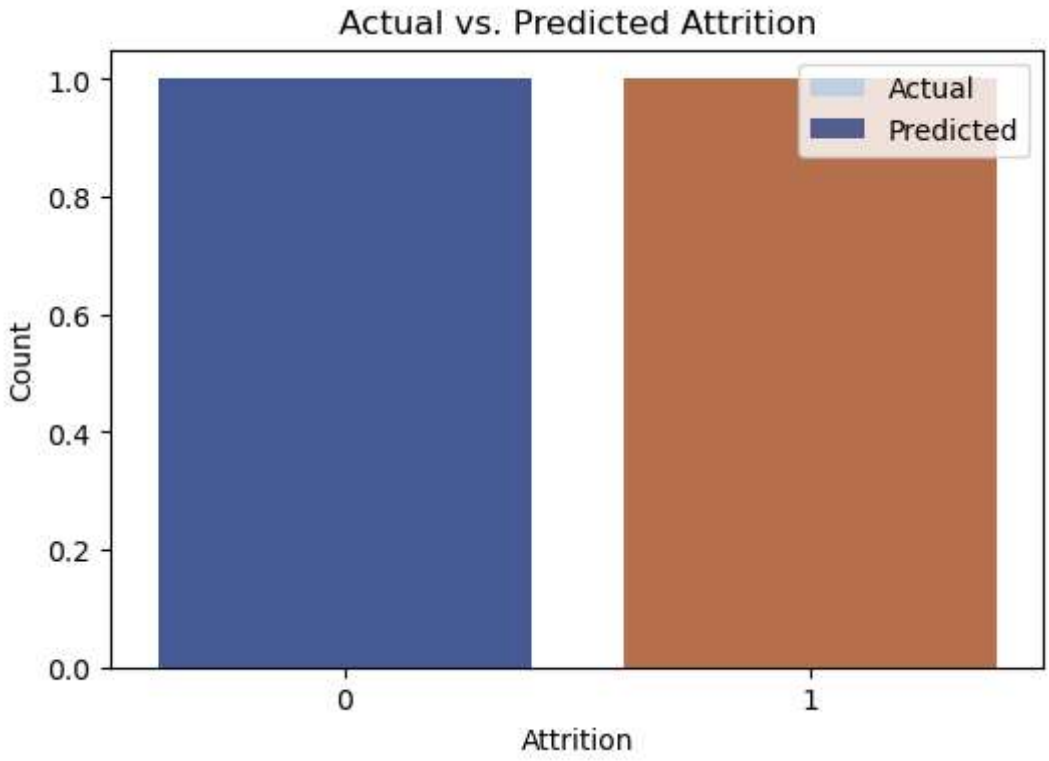
Classification

In [40]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_r

# Sample data (replace this with your dataset)
data = {
    'YearsAtCompany': [5, 3, 7, 2, 4, 8, 6, 1, 3, 2],
    'YearsInCurrentRole': [2, 1, 3, 1, 2, 3, 2, 1, 1, 2],
    'YearsSinceLastPromotion': [1, 2, 3, 1, 2, 3, 2, 1, 1, 2],
    'YearsWithCurrManager': [3, 1, 5, 1, 2, 4, 3, 1, 2, 3],
    'Attrition': [0, 1, 1, 0, 0, 1, 1, 0, 0, 1]  # 0: No Attrition, 1: Attriti
}

# Create DataFrame
df = pd.DataFrame(data)

# Split the data into features (X) and target variable (y)
X = df[['YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion', 'Ye
y = df['Attrition']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

# Create and fit the Decision Tree classifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)

# Make predictions on the test set
y_pred = decision_tree.predict(X_test)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)

# Plot the Decision Tree
from sklearn.tree import plot_tree

plt.figure(figsize=(10, 8))
plot_tree(decision_tree, feature_names=X.columns, class_names=["No Attrition",
plt.show()

# Plot a bar chart of the actual vs. predicted values
plt.figure(figsize=(6, 4))
sns.countplot(x=y_test, palette="pastel", alpha=0.7, label='Actual')
sns.countplot(x=y_pred, palette="dark", alpha=0.7, label='Predicted')
plt.xlabel('Attrition')
```

```
plt.ylabel('Count')
plt.title('Actual vs. Predicted Attrition')
plt.legend()
plt.show()
```

```
Accuracy: 1.0
Confusion Matrix:
[[1 0]
 [0 1]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```





Check the Accuracy of model

In [41]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_r

# Sample data (replace this with your dataset)
data = {
    'YearsAtCompany': [5,3,7,2,4,8,6,1,3,2],
    'YearsInCurrentRole': [2,1,3,1,2,3,2,1,1,2],
    'YearsSinceLastPromotion': [1,2,3,1,2,3,2,1,1,2],
    'YearsWithCurrManager': [3,1,5,1,2,4,3,1,2,3],
    'Attrition': [0, 1, 1, 0, 0, 1, 1, 0, 0, 1]  # 0: No Attrition, 1: Attriti
}

# Create DataFrame
df = pd.DataFrame(data)

# Split the data into features (X) and target variable (y)
X = df[['YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion', 'Ye
y = df['Attrition']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

# Create and fit the logistic regression model
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = logistic_model.predict(X_test)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
```

```
Accuracy: 1.0
Confusion Matrix:
[[1 0]
 [0 1]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

Thank You!

In [ ]: